Nothing Found

Your search for **+(natural +<near/5> +language) +<and> +(concept* +<near/5> +graph*)** did not return any results.

You may want to try an Advanced Search for additional options.

Please review the Quick Tips below or for more information see the Search Tips.

Quick Tips

- Enter your search terms in lower case with a space between the terms.

  ```
  sales offices
  ```

  You can also enter a full question or concept in plain language.

  ```
  Where are the sales offices?
  ```

- Capitalize proper nouns to search for specific people, places, or products.

  ```
  John Colter, Netscape Navigator
  ```

- Enclose a phrase in double quotes to search for that exact phrase.

  ```
  "museum of natural history" "museum of modern art"
  ```

- Narrow your searches by using a + if a search term must appear on a page.

  ```
  museum +art
  ```

- Exclude pages by using a - if a search term must not appear on a page.

  ```
  museum -Paris
  ```

Combine these techniques to create a specific search query. The better your description of the information you want, the more relevant your results will be.

```
museum +"natural history" dinosaur -Chicago
```

Your search matched **16** of **1117582** documents.
A maximum of **500** results are displayed, **15** to a page, sorted by **Relevance** in **Descending** order.

Refine This Search:
You may refine your search by editing the current search expression or entering new one in the text box.

(natural <near/5> language) <and> (concept* <near/5> Search

☐ Check to search within this result set

Results Key:
JNL = Journal or Magazine **CNF** = Conference **STD** = Standard

---

1 **Capture, integration, and analysis of digital system requirements with conceptual graphs**
Cyre, W.R.;
Knowledge and Data Engineering, IEEE Transactions on , Volume: 9 , Issue: 1 , Jan.-Feb. 1997
Pages:8 - 23

[Abstract] [PDF Full-Text (192 KB)] **IEEE JNL**

---

2 **Conceptual representation of waveforms for temporal reasoning**
Cyre, W.R.;
Computers, IEEE Transactions on , Volume: 43 , Issue: 2 , Feb. 1994
Pages:186 - 200

[Abstract] [PDF Full-Text (1228 KB)] **IEEE JNL**

---

3 **Acquiring and managing knowledge using a conceptual structures approach: introduction and framework**
Berg-Cross, G.; Price, M.E.;
Systems, Man and Cybernetics, IEEE Transactions on , Volume: 19 , Issue: 3 , May-June 1989
Pages:513 - 527

[Abstract] [PDF Full-Text (1432 KB)] **IEEE JNL**

---

4 **Advanced animation framework for virtual character within the MPEG-standard**
Preda, M.; Preteux, F.;
Image Processing. 2002. Proceedings. 2002 International Conference on , Volum 3 , 24-28 June 2002

Pages:509 - 512 vol.3

[Abstract] [PDF Full-Text (377 KB)] IEEE CNF

---

5 **Reasoning on aspectual-temporal information in French within conceptual graphs**
Amghar, T.; Battistelli, D.; Charnois, T.;
Tools with Artificial Intelligence, 2002. (ICTAI 2002). Proceedings. 14th IEEE International Conference on , 4-6 Nov. 2002
Pages:315 - 322

[Abstract] [PDF Full-Text (279 KB)] IEEE CNF

---

6 **A concept graph based confidence measure**
Hacioglu, K.; Ward, W.;
Acoustics, Speech, and Signal Processing, 2002. Proceedings. (ICASSP '02). IEEE International Conference on , Volume: 1 , 13-17 May 2002
Pages:I-225 - I-228 vol.1

[Abstract] [PDF Full-Text (335 KB)] IEEE CNF

---

7 **The Chronological Information Extraction System (CHESS)**
O'Neil, P.; Woojin Paik;
Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on , Volume: 2 , 11-14 Oct. 1998
Pages:1674 - 1679 vol.2

[Abstract] [PDF Full-Text (696 KB)] IEEE CNF

---

8 **Visual feedback for validation of informal specifications**
Thakar, A.; Cyre, W.;
Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 1994., MASCOTS '94., Proceedings of the Second International Workshop on , 31 Jan.-2 Feb. 1994
Pages:411 - 412

[Abstract] [PDF Full-Text (164 KB)] IEEE CNF

---

9 **Conceptual graph-based synthesis of robotic assembly operations**
Kapitanovsky, A.; Maimon, O.;
Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on , 12-14 May 1992
Pages:2413 - 2418 vol.3

[Abstract] [PDF Full-Text (432 KB)] IEEE CNF

---

10 **Mapping design knowledge from multiple representations**
Cyre, W.;
Computer Design: VLSI in Computers and Processors, 1991. ICCD '91. Proceedings., 1991 IEEE International Conference on , 14-16 Oct. 1991
Pages:406 - 409

[Abstract] [PDF Full-Text (328 KB)] IEEE CNF

---

11 **A semantic interpreter for a transportable command language interfa**
Moore, L.E.; O'Neal, M.B.;
Applied Computing, 1990., Proceedings of the 1990 Symposium on , 5-6 April 19

Pages:202 - 208

[Abstract] [PDF Full-Text (392 KB)] **IEEE CNF**

---

12 **A graph based knowledge retrieval system**
Kamel, M.; Quintana, Y.;
Systems, Man and Cybernetics, 1990. Conference Proceedings., IEEE Internation
Conference on , 4-7 Nov. 1990
Pages:269 - 275

[Abstract] [PDF Full-Text (656 KB)] **IEEE CNF**

---

13 **A linguistic model for computer graphics**
Tokuta, A.; Staudhammer, J.;
Computers and Communications, 1988. Conference Proceedings., Seventh Annu
International Phoenix Conference on , 16-18 March 1988
Pages:262 - 268

[Abstract] [PDF Full-Text (448 KB)] **IEEE CNF**

---

14 **The Analyst Assist project**
Pyburn, R.;
Requirements Capture and Specification for Critical Systems, IEE Colloquium on , 24 Nov 1989
Pages:5/1 - 5/2

[Abstract] [PDF Full-Text (100 KB)] **IEE CNF**

---

15 **A process model for unifying systems engineering and project management**
Boardman, J.T.;
Engineering Management Journal , Volume: 4 , Issue: 1 , Feb. 1994
Pages:25 - 35

[Abstract] [PDF Full-Text (540 KB)] **IEE JNL**

---

1 2 Next

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	4	((semantic near3 (structure or relation)) same (conceptual near3 graph)) and index$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/01/18 16:59
L2	5	((((morpholog$$4 with failure) and (natural same language)) and semantic) and node) and (level or tree)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/01/18 16:57

A Semantic Interpreter for a Transportable Command Language Interface

Lisa E. Moore, Grambling State University
Micheal B. O'Neal, Louisiana Tech University

Abstract

This paper describes the implementation of a semantic interpreter for a transportable command language interface that accepts natural language input. Included in the discussion are the theories of representation for semantic knowledge that influenced the implementation and a functional description of the interpreter.

Introduction

Semantic interpretation is the process of generating a structure that represents the meaning of a sentence[19]. The computer can be said to "understand" the structure if it can incorporate a declarative sentence into its database and use it for future inference, or perform a requested action for a command, or make an appropriate reply to a question. The generated structure must therefore conform to an internal representation that may be used to bring about these actions. The operation of a semantic interpreter is based on two underlying assumptions: 1) that syntax is an encoding of meaning and 2) that the meaning of the whole (i.e., the sentence) is the sum of the meaning of its parts. The latter assumption is called compositionality[1].

Three theories of representation for semantic knowledge have influenced the design of the semantic interpreter. They are semantic nets by Quillian[1], conceptual dependency by Roger Schank[30], and conceptual graphs by John Sowa[35]. A brief overview of each formalism and a discussion of their combined influence on the semantic interpreter is presented below.

Semantic Nets

Semantic nets[1][37] were first proposed as a model of human cognition in keeping with the school of AI theorists who believe that computers would become intelligent by doing things the way humans do them.

Semantic nets are graphically represented as nodes connected by arcs. Both the nodes and the arcs are labeled. Traditionally, the nodes represent objects and the arcs represent the relationships between the objects. The labels are not standardized and may be designed by the user depending on the information to be represented.

Figure 1 is a representation of the statement "Diane is a mother."

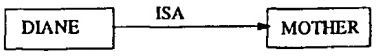


Figure 1 Semantic Network A

The major advantage of semantic nets is the ability of subclasses to inherit qualities from superclasses[1]. Figure 2 shows an example.

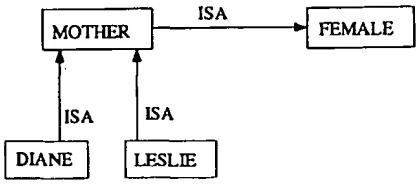


Figure 2 Semantic Network B

The fact that a mother is female is succinctly captured. Diane and Leslie inherit female-ness from their superclass "MOTHER." A system whose database contained the above fact, could affirmatively answer the query "Is Leslie female" by following the ISA links.

A problem with semantic nets is the difficulty in distinguishing between individuals and the class to which they belong when dealing with inheritance.

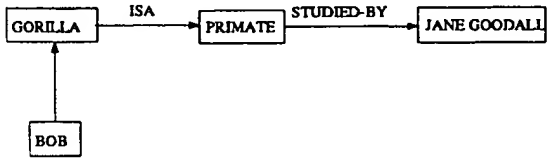


Figure 3 Semantic Network C

In figure 3, a gorilla is a primate, and Jane Goodall studies primates. Bob, however, is a specific gorilla and therefore also a primate, but he may not be a subject of study by Jane Goodall.

A solution to this problem is to differentiate between concepts and instances of a concept. Links from a concept should be true for that concept and validly inheritable by all instances of that concept. Links from an instance of a concept are used to describe qualities specific to the individual. A better version of the previous example is given in figure 4.

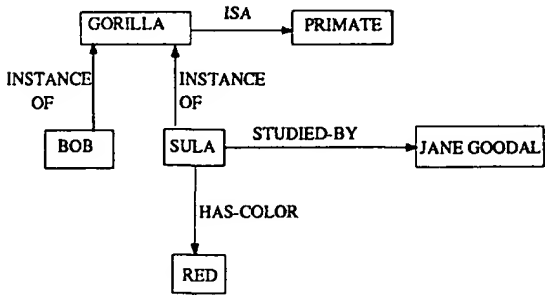


Figure 4 Semantic Network D

The combination of two nodes and their intervening link is clearly another representation of binary predicates, e.g., ISA(BOB,GORILLA).

One of the earliest examples of the use of a version of semantic nets in an understanding system was SIR: Semantic Information Retrieval by Bertram Raphael[1] in which "understanding" was demonstrated by the computer enganging in conversations that required awareness of the meaning of the topic. Raphael used binary predicates like part_of(finger,hand) and the attributes subset and superset.

More recently YUCCA II, a UNIX consultation facility[18] has made use of an object database using a generalized hierarchy with property inheritance and distinction between individual instances and object classes.

Deliyanni[11] defines an extended version of semantic nets that gives them the expressive power of predicate logic and can also be used as an abstract data structure. In Deliyanni's formalism, terms are represented by nodes, binary predicates are arc labels and an atom consists of an arc and its two end nodes. The arc direction gives the order of the arguments of the predicates. Figure 5 below shows the extended semantic net representing that every person is either male or female.

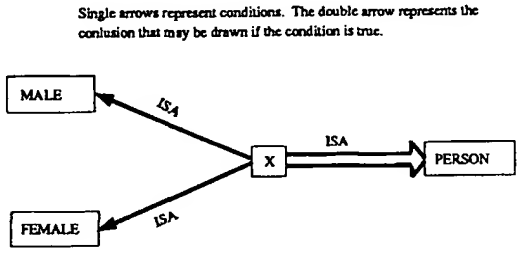


Figure 5 Extended Semantic Net

Conceptual Dependency

Roger Schank[1][37] introduced the idea of conceptual dependency. Conceptual dependency is a representation based on semantic primitives. Schank believes that for a given concept the representation should be unique and unambiguous. Notice that we are dealing with concepts not sentences. The idea of uniqueness of a concept goes much farther that recognizing the relationship between passive and active statements of the same thought. All of the sentences given below should have the same representation.

He owns that car.
That car is his.
That car belongs to him.
He is the owner of that car.

Schank associates this sameness of representation with the human ability to paraphrase.

Schank defined a number of primitive acts, such as the now familiar ATRANS and PTRANS, which were intended to be used as building blocks for concept representation.

Conceptual dependency has two advantages over semantic nets as a knowledge representation:

1. The primitive acts themselves represent several inferences.
2. Fewer rules are needed.

An example is the conceptual dependency representation of "The gorilla ate the banana."

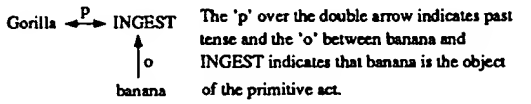


Figure 6 The INGEST primitive

By the definition of the INGEST primitive we may infer that the location of the banana is now inside the gorilla and that the banana no longer exists outside of the gorilla. A rule-based system would require three explicit rules to represent the concepts captured by this single primitive.

Schank developed a system called MARGIE to implement his theories about conceptual dependency[1].

Conceptual Graphs

Conceptual Graphs are a representation formalism for information about things and events as well as semantic information. They were introduced by John Sowa[36][35].

A conceptual graph contains three types of objects: concept types, conceptual relations and referents. A concept is constrained to a particular value by assigning a referent.

[CITY: {Ruston, Louisiana}]

In the above example CITY is a concept whose referent is Ruston, Louisiana. Concepts are linked by conceptual relations to form conceptual graphs. In figure 7, Bob the gorilla is the agent of the action concept "ATE" and the concept "BANANA" is the object.

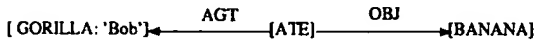


Figure 7 Conceptual graph of "Bob the gorilla ate the banana"

The rules of formation for conceptual graphs are syntactic[12]. Concept types may be assigned attributes using the relationship "<." Thus GORILLA < PRIMATE assigns to all instances of GORILLA the attribute of PRIMATE. This allows us to infer that Bob from the previous example is a primate. Several algorithmic operations are defined on conceptual graphs such as join, contraction, and matching. Conceptual Graphs obey the theory of compositionality and also allow inference.

Sowa has implemented a semantic interpreter using conceptual graphs based on the following algorithm[36]:

1. Look up each word in a lexicon of graphs that represent the ways concepts and relations can be linked.
2. Build the representation of the input sentence by joining the graphs for each word in the sentence.

The syntactic parse tree that is the input to the semantic interpreter guides the joining of the graphs. Syntactic ambiguities are resolved because the semantic information stored in the graphs rejects illegal joins.

Influences on the Design of the Semantic Interpreter

During the development of the semantic interpreter, each word was initially assigned to a category, for example, ACTION(FORMAT), OBJECT(DISK). This method required a process by which the two would be linked to show a valid concept, and equally as important, a method of determining which concepts were not valid. The knowledge base might also contain the entries OBJECT(FILES), ACTION(LIST). A representation capable of showing that FILES is a valid object of LIST but not of FORMAT, while DISK is a valid object for both, was needed. Referring to Schank's theory that each concept should have a unique and unambiguous representation, it was deemed desirable to use a representation similar to Schanks primitives, while maintaining the readability of semantic nets. The classic primitives as defined by Schank are singularly unsuited to use over the application domain. I decided to take the clausal representation of semantic nets, to collapse it, and to combine related categories into primitives. These primitives were then used as cases during syntactic parsing. Using the four words given above: LIST, FORMAT, DISK, and FILES, their relationship can be captured by the semantic net shown in figure 8.

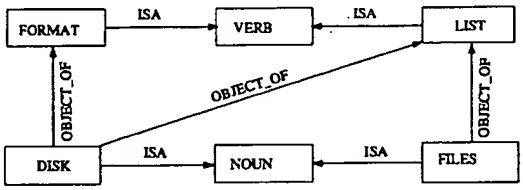


Figure 8 Clausal form of a Semantic Network

Representing the same information in clausal form we have

ISA(FORMAT,VERB)
ISA(LIST,VERB)
ISA(DISK,NOUN)
ISA(FILES,NOUN)
OBJECT_OF(FILES,LIST)
OBJECT_OF(DISK,LIST)
OBJECT_OF(DISK,FORMAT)

Collapsing the predicates creates

VERB(FORMAT)
VERB(LIST)
NOUN(DISK)

NOUN(FILES)
OBJECT_OF(FILES,LIST)
OBJECT_OF(DISK,LIST)
OBJECT_OF(DISK,FORMAT)

By combining the predicate we derive

VERB_OBJ(LIST,FILES)
VERB_OBJ(LIST,DISK)
VERB_OBJ(FORMAT,DISK)

The verb_obj primitive represents that its first argument is a verb and its second argument is a value which is a valid object of the verb's action. In the implementation I have renamed the verb_obj primitive to act_obj.

The semantic knowledgebase is composed of all possible valid relationships between the words which are recognized by the syntactic parser. The relationships are described by cases.

The algorithm by which the semantic interpreter performs its function is based on Sowa's conceptual graphs theory. Sowa has implemented a semantic interpreter by joining graphs of word meaning to create a structure that captures a concept. The semantic interpreter of this project operates on the same theory. The parse that is sent to the semantic interpreter by the syntactic parser is a list of cases. These cases are a clausal representation of graphs. The cases, when combined, form a representation of the meaning of the sentence that is a composite of the meaning of the individual cases. The Semantic Interpreter checks each case for semantic correctness. If all the cases that represent a sentence are correct, then the concept expressed by the sentence is semantically valid. For example, the sentence "list all files" will generate the following case representation.

action("list")
act_obj("list","files")
adj("all")

This sentence is semantically correct if the semantic database contains an entry that matches each case.

Implementation of the Semantic Interpreter

The semantic interpreter (SI) accepts the output of a syntactic parser and determines if it makes sense. The most important feature of the SI is its domain specificity. The sentence "TELL THE TRUTH" is well formed and meaningful in the real world but makes no sense in the microworld of command languages. When I speak of "making sense" with respect to the SI, I mean that the input is meaningful over the application domain.

The SI has two parts, the lexicon, which contains domain specific information about the concepts represented by words and the roles that those words may play in a "correct" sentence, and the interpreter proper.

The interpreter moves through the structure generated by the parser and for each element in that structure, determines if it is conceptually correct by referring to the lexicon. The sentence will already have been checked for syntactic correctness, so the interpreter is only checking for semantic correctness.

The output of the interpreter will be the same structure as was passed to it by the parser if that structure is determined to be correct or a flag indicating no correct parses.

Functional Description

The semantic interpreter is implemented in keeping with the cognitive theory that humans understand infinitely many different sentences because we parse them syntactically and derive the meaning of the sentence from the combined meaning of its parts.

The semantic interpreter (SI) is called by an interface driver with no parameters. The parse(s) which are passed to the interpreter are generated by an ATN parser . Each parse consists of a list of cases that describe the relationship of the words in the input sentence. The SI recursively removes each case from the list and checks its validity. A case, representing some portion of the input sentence is valid if it corresponds to an entry in the database or if it fits a predefined pattern. A sentence is semantically correct if all the case structures of which its representation are composed, are valid.

Patterns are used for situations in which user defined names may be encountered. For example "CHANGE DIRECTORY TO C:\USR\PROCESS." It is not possible to predict the names which users will assign to their directories and/or files, nor is is possible to place any such entry into the semantic knowledgebase. This difficulty has been sloved by using pattern matching. For the example given above, the following predicate constitutes a pattern by which the preposition "TO" and its object "C:\USR\PROCESS" may be recognized as semantically correct.

```
is-pattern(X):-
   X = (prep_obj("to",Z)),
  part_speech(Z,pathname,Z).
```

A tokenizer/preprocessor is used to determine that C:\USR\PROCESS is a pathname and to assert a fact to that effect into the database.

As an example of the operation of the SI assume that we have, in the knowledgebase for the semantic interpreter, the following entries:

ACT_OBJ(FORMAT,DISK)
ACT_OBJ(LIST,FILES)
MODT(THE)
MODT(ALL)

As input to the command language interface we give "format the files" which will be transformed by the syntactic parser to [act_obj(format,files),modt(the)]. This request is not semantically valid because "files" is not a valid receiver of the action of the verb "format." The inappropriateness of the request is indicated by the absence of an [act_obj(format,files)] entry in the database. Alternatively, "list all files" and "format the disk" are valid semantic requests.

Differences In My Approach

In previous implementations of semantic interpreters, the approach has been to take the syntactic parse tree, which consisted solely of the parts of speech of the words in the sentence, and to try to assign meaning to the order of the words. The system that I have designed deviates from this approach in that the parse tree produced by the syntactic parser is itself a representation of sentence meaning. Moreover, I have based this implementation on two significant assumptions:

1. That over a well bounded domain, it is possible to determine the underlying meaning of a sentence through syntactic analysis. If this is true, from syntax alone it should be possible to derive a valid representation of sentence meaning. By concept, I mean the underlying idea of a sentence rather than the actual content. For example, Given the sentence "List the files", the concept is that an action is to be performed on some object. The content of the sentence gives definition to the action and its object. Using the concept of the sentence, a semantic representation can be built by filling in the "content" over the structure of the concept. The concept is

   ACTION(X)
   OBJECT(Y)

   Filling in the content, X = LIST and Y = FILES. Combining the two into a valid case gives ACT_OBJ("LIST","FILES") which represents the meaning of the sentence.
2. Perhaps we perceive the world as a set of microworlds. Each thing considered in its proper context is the basis for understanding.

Additionally, the SI which I have implemented differs from other systems in the following two ways:

1. No manipulation of the syntactic parse tree is required to determine semantic validity.
2. Previous systems have allowed the ad-hoc creation of database objects to compensate for unpredictable events. I have used patterns.

Summary

The semantic interpreter for the transportable command language interface was implemented based on Schanks theory that sentences expressing the same concept should have the same representation. The SI incorporates three key ideas:

1. The use of binary predicates to represent semantically valid relationships.
2. The use of case structures for both syntactic and semantic analysis based on the assumption that over a well bounded domain sentence structure is a valid indicator of the concept represented by the sentence.
3. The use of generic patters for word/concepts not "known" by the system.

REFERENCES

[1] Barr, Avron and Edward A. Feigenbaum, editors. *The Handbok of Artificial Intelligence*. William Kaufman, Los Altos, CA., 1981.

[2] Bates, Madeleine. The theory and practice of augmented transition network grammars. In Leonard Bolc, editor, *Natural Language Communication with Computers*, pages 191–260. Springer-Verlag, New York, 1978.

[3] Bolc, Leonard, Adam Kowalski, Malgorzata Kozlowska, and Tomasz Strzalkowski. A natural language information retrieval system with extensions towards fuzzy reasoning. *Intl. J. Man-Machine Studies*, 23(4):335–367, October 1985.

[4] Bratko, Ivan. *Prolog Programming for Artificial Intelligence*. Addison-Wesley, Reading, MA, 1987.

[5] Chomsky, Noam. *Syntatic Strucutures*. Mouton and Co., The Hague, 1964.

[6] Chomsky, Noam. *Studies on Semantics in Generative Grammars*. Mouton and Co., The Hague, 1975.

[7] Christopher, Jay and Rodger Knaus. Frames in prolog. *AI Expert*, 4(3):19–24, March 1989.

[8] Clinger, Barbara. Definite clause grammars in turbo prolog. *Turbo Technix*, 1(6):80–84, Sept/Oct 1988.

[9] Clocksin, W.F. and C.S. Mellish. *Programming in Prolog*. Springer-Verlag, New York, 1984.

[10] Cohen, Daniel I.A. *Introduction to Computer Theory*. John Wiley and Sons, Inc., New York, 1986.

[11] Deliyanni, Amaryllis and Robert A. Kowalski. Logic and semantic networks. *Comm. ACM*, 22(3):184–192, March 1979.

[12] Fargues, Jean, Marie-Claude Landau, Anne Dugourd, and Laurent Catach. Conceptual graphs. *IBM J. of Res. and Dev.*, 30(1):70–79, January 1986.

[13] Fikes, Richard and Tom Kehler. The role of frame based representation in reasoning. *Comm. ACM*, 28(9):904–919, September 1985.

[14] Frost, R. and J. Launchbury. Constructing natural language interpreters in a lazy functional language. *The Computer Journal*, 32(2):108–121, April 1989.

[15] Gosse, Bouma, Esther Konig, and Hans Uszkoreit. A flexible graph unification formalism and its application to natural language processing. *IBM J. of Res. and Dev.*, 32(2):170–184, March 1988.

[16] Grosz, B.J. Team: A transportable natural language interface system. In *Proc. 1983 Conf. on Applied Natural Language Processing*, pages 39–45, 1983.

[17] Hardy, Jr., Trotter. The syntax of interactive command languages: A framework for design. *Software Practice and Experience*, 12:67–75, 1982.

[18] Hegner, Stephen J. A representation of command language behavior for an operating system consultation facility. In *Proc. 4th Conf. on AI Applications*, pages 50–55, 1988.

[19] Hirst, Graeme. Semantic interpretation and ambiguity. *Artificial Intelligence*, 34(2):137–177, March 1988.

[20] Jackendoff, Ray S. *Semantic Interpretation in Generative Grammars*. MIT Press, Cambridge, MA., 1972.

[21] Jacobs, Paul S. Knowledge intensive natural language generation. *Artificial Intelligence*, 33(3):325–378, November 1987.

[22] Kaiser, Gail E. Automatic extension of an atn knowledge base. *Comm. ACM*, 24(9):587–593, September 1981.

[23] Koutsoudas, Andreas. *Writing Transformational Grammars*. McGraw Hill, New York, 1966.

[24] Lieber, Justin. *Noam Chomsky: A Philosophical Overview*. G.K. Hall and Co., 1975.

[25] McCalla, Gordon I. and Jeffrey R. Sampson. Muse: A model to understand simple english. *Comm. ACM*, 15(1):29–40, January 1972.

[26] McKevitt, P. and Y. Wilks. Transfer semantics in an operating system consultant: The formalization of actions involving object transfer. In *Proc. Tenth IJCAI*, pages 569–1059, Milan, 1987.

[27] Moran, Thomas P. The command language grammar: A representation for the user interface of interactive computer systems. Technical report, Xerox Palo Alto Research Center, Palo Alto, CA, 1981.

[28] Oden, Gregg C. On the use of semantic constraints in guiding syntatic analysis. *Intl. J. Man-Machine Studies*, 19(4):335–357, October 1983.

[29] Sager, Naomi. *Natural Language Information Processing*. Addison–Wesley, Reading, MA., 1981.

[30] Schank, Roger. *The Cognitive Computer*. Addison-Wesley, Reading, MA., 1984.

[31] Schneiderman, Ben. *Designing the User Interface*. Addison-Wesley, Reading, MA., 1987.

[32] Simmons, Robert F. Man-machine interfaces: Can they guess what you want. *IEEE Expert*, Spring 1986.

[33] Simmons, Robert F. and Daniel Chester. Relating sentences and semantic networks with procedural logic. *Comm. ACM.*, 25(8):527–547, August 1982.

[34] Smith, Howard R., Warren H. Harris, and Dan Simmons. Frameworks: A uniform approach to knowledge representation for natural language processing. Technical report, United Technologies Research Center, East Hartford, CT 06108, 1987.

[35] Sowa, John F. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, Reading, MA, 1984.

[36] Sowa, John F. and Eileen C. Way. Implementing a semantic interpreter using conceptual graphs. *IBM J. of Res. and Dev.*, 30(1):57–69, January 1986.

[37] Tanimoto, Steven L. *The Elements of Artificial Intelligence*. Computer Science Press, Inc., Rockville, MD., 1987.

[38] Thompson, Craig W. *Using Menu-Based Natural Language Understanding to Avoid Problems Associated with Traditional Natural Language Interfaces to Databases*. PhD thesis, Univ. of Texas, Austin, 1984.

[39] Weiskamp, Keith and Terry Hengl. *Artificial Intelligence Programming with Turbo Prolog*. John Wiley and Sons, Inc., New York, 1988.

[40] Wilensky, Robert, Yigal Arens, and David Chin. Talking to unix in english: An overview of uc. *Comm. of the ACM*, 27(6):575–593, June 1984.

[41] Winograd, Terry. *Language as a Cognitive Process*. Addison-Wesley, Reading, MA., 1983.

[42] Winston, Patrick Henry. *Artificial Intellligence*. Addison-Wesley, Reading, MA., 1977.

[43] Woods, William A. Transition network grammars for natural language analysis. *Comm. ACM*, 13(10):591–606, October 1970.

[44] Young, S.J. and C. Proctor. Ufl: An experimental frame language based on abstract data types. *The Computer Journal*, 29(4):340–347, August 1986.

A CONCEPT GRAPH BASED CONFIDENCE MEASURE

Kadri Hacioglu and Wayne Ward

Center for Spoken Language Research
University of Colorado at Boulder
E-mail: {hacioglu,whw}@cslr.colorado.edu

ABSTRACT

In this paper, the confidence measure of a hypothesized word is derived from its posterior probability. In contrast to common approaches, in which N-best lists or word graphs/lattices are used, the posterior probabilities are derived from a concept graph. The concept graph is obtained from a word graph through a partial parsing process using semantic grammars. This approach allows us to use relatively complex and better language models along with acoustic models to compute word posterior probabilities. The language model used is comprised of stochastic context free grammars (one for each concept) and an n-gram concept language model. We show that the posterior probabilities computed on concept graphs outperform those computed on word graphs when used as confidence measures. Results are presented within the context of Colorado University (CU) Communicator System; a telephone-based dialog system for making travel plans by accessing information about flights, hotels and car rentals.

1. INTRODUCTION

In several tasks the output of the speech recognizer is far from perfect. This creates problems in applications that use the transcription directly. For instance, in a dialog system, errors in the output word sequence can lead to a dialog flow that diverges from the user's goal. This results in a longer, or maybe unsuccessful, dialog leaving the user confused, frustrated and dissatisfied with the interaction. So, it is very important for a dialog manager to spot incorrectly recognized words and act accordingly to achive human-like performance. Another example is the unsupervised adaptation of the acoustic models using maximum likelihood linear regression (MLLR). Adaptation with an incorrect transcription degrades the performance. Therefore, one needs a method to spot incorrect words and exclude them from adaptation.

Confidence measures are used to label the words at the output of the speech recognizer as correct or incorrect. The basic approach is to select a set of effective features and find a way of combining them into a confidence measure [1, 2, 3]. Although the combination of several features improves the performance, it is usually not much better than that of the best feature.

The word posterior probabilities have been proposed and used as a confidence measure or as an additional feature [4, 5, 6]. The utility of the word posterior probabilities for confidence annotation has been clearly illustrated in [7]. Those probabilites were calculated using either N-best lists or word graphs/lattices. The knowledge sources were acoustic models and n-gram language models. Recently, in dialog systems, it has been observed that additional features at higher levels, e.g. parsing or understanding levels, improve the performance significantly [8, 9] .

In this paper, we incorporate knowledge at the understanding level as a statistical language model (SLM). The SLM has been developed within a flexible speech understanding framework [10]. It consists of a set of stochastic context free grammars, one for each concept, and dialog context conditioned trigram concept LMs. We use SCFGs to parse the word graph into a concept graph. Then, we compute word posterior probabilities on the concept graph using acoustic models, SCFGs, and concept trigram LMs interpolated with word/class based trigram LM. We compare a confidence measure computed on concept graphs to that computed on word graphs. We provide results that show the better performance of the concept graph based confidence measure.

The paper is organized as follows. In section 2, we summarize the posterior probability computation on word graphs using the forward/backward algorithm. The extension of this method to concept graphs is explained in section 3. The experimental setup and results are presented in section 4. The last section includes concluding remarks and possible future work.

2. WORD PROBABILITIES ON WORD GRAPHS

In this section, we present a forward/backward type algorithm for calculating the word probabilities on word graphs similar to that presented in [7]. Let s be the start frame and e be the end frame of a word w in a sequence of words w_1^N that spans a time interval of length T frames. We define a word event as $[w, s, e]$. We are interested in the probability of this event given the acoustic observation o_1^T. This probability should be calculated over all possible word sequences that contain w at the interval [s, e]. However, it is a common practice to restrict the computation to a word graph, as it is a compact representation of the most probable word sequences.

We first explain the word graph generated by the speech recognizer (CMU-Sphinx II [11]). It is a directed weighted acyclic graph. It is built from the word lattice created during frame synchronous tree lexicon Viterbi beam search. Its nodes represent unique (w, s) pairs. Edges are labeled with end frames and acoustic model scores, $p(o_s^e/w)$. They point to nodes $(e+1, w_s)$, where e is the end frame of the word w and w_s is its successor. Note that

The work is supported by DARPA through SPAWAR under grant #N66001-00-2-8906.

0-7803-7402-9/02/$17.00 ©2002 IEEE

each unique node can link to more than one node, thanks to the possibility of more than one end time and successor of a particular word that starts at frame s. Each path through the word graph is a word sequence that spans the time interval of length T. The set of all paths defines the ensemble on which we calculate the posterior probabilities.

Each word event $[w, s, e]$ corresponds to a set of edges in the word graph. So, the probability of a word event is the total probability of all edges associated with it. In developing the forward/backward type algorithm, we consider the edges as HMM like states . The emission probabilities are the acoustic scores kept at the edges. The transition probabilities are provided by the language model in use, which is assumed to be a word based trigram LM in the sequel. Its extension to a class-based trigram LM is straightforward. To derive the algorithm we need to define edges uniquely. So, the edge from node (w_i, s) to $(w_{i+1}, e+1)$ is defined as $E^{w_{i+1},e+1}_{w_i,s}$. We define the forward probability of an edge as $\alpha(E^{w_{i+1},e+1}_{w_i,s})$. The following recursion can be used to compute the forward probabilities:

$$\alpha(E^{w_{i+1},e+1}_{w_i,s}) = P(o_s^e|w_i) \cdot \sum_{w_{i-1},s'} \alpha(E^{w_i,s}_{w_{i-1},s'}) \cdot P(w_{i+1}|w_{i-1}, w_i) \quad (1)$$

Similarly, we define the backward probability of an edge as $\beta(E^{w_{i+1},e+1}_{w_i,s})$. The recursion for the backward probabilities is

$$\beta(E^{w_{i+1},e+1}_{w_i,s}) = \sum_{w_{i+2},e'+1} \beta(E^{w_{i+2},e'+1}_{w_{i+1},e+1}) \cdot P(o^{e'}_{e+1}|w_{i+1}) \cdot p(w_{i+2}/w_i, w_{i+1}) \quad (2)$$

Once we have computed the forward-backward probabilities we can calculate the edge posterior probabilities, and in turn, the word posteriors. The posterior probability of an edge can be obtained as

$$p(E^{w_{i+1},e+1}_{w_i,s}) = \frac{\alpha(E^{w_{i+1},e+1}_{w_i,s}) \cdot \beta(E^{w_{i+1},e+1}_{w_i,s})}{p(o_1^T)} \quad (3)$$

where $p(o_1^T)$ can be obtained as the summation of the forward probabilities of the edges that end at frame T. Equivalently, it can be calculated as the summation of the backward probabilities of the edges that start at the initial frame.

In fact, the probability in (3) is the posterior probability of the word that starts at frame s, ends at frame e and precedes the word w_{i+1}. In [7], it has been demonstrated that the use of this probability as a confidence measure on a hypothesized word does not give satisfactory results. Instead, we use

$$C(w_i) = \sum_{w_{i+1},e+1} p(E^{w_{i+1},e+1}_{w_i,s}) \quad (4)$$

This amounts to the summation of the probabilities of the edges outgoing from node (w_i, s). For other possibilities of confidence measures derived from posterior probabilities the reader is referred to [7].

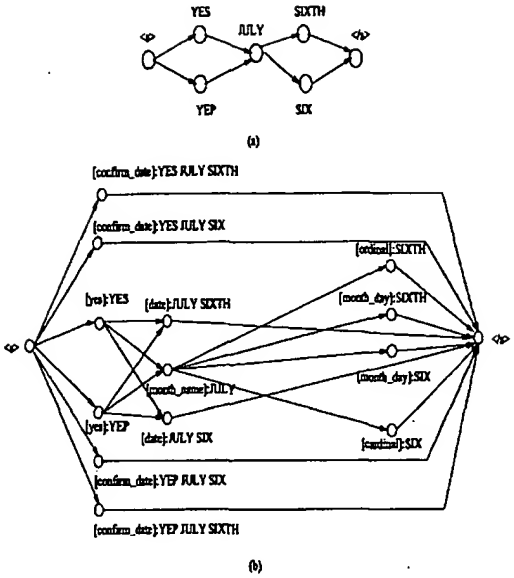


Fig. 1. (a) word graph ,(b) concept graph

3. CONCEPT/WORD PROBABILITIES ON CONCEPT GRAPHS

The structure of the concept graph is same as the word graph. The nodes are associated with (c, s) pairs. The concept c can span a number of words. An example of a concept graph derived from a word graph by partial parsing using a semantic grammar is depicted in Figure 1.

The concepts are classes of phrases with the same meaning. That is, a concept class is a set of phrases that can be used to express that concept (e.g. [date], [yes]). Each concept (except degenerate single word concepts) is written as a context free grammar (CFG) and compiled into a recursive transition network (RTN). The arcs of RTNs are populated with probabilities using a training method based on simple counting and smoothing. So, the multiplication of the arc probabilities that traverse a phrase gives the probability of that phrase. The concept patterns are modeled by n-gram LMs conditioned on the dialog context. The dialog context has been taken as the system's last prompt. A detailed discussion of these LMs can be found in [10].

Similar to the word event defined in the preceding section, we define a concept event, $[c, s, e]$, which can also be associated with a set of edges outgoing from node (c, s). Here, the start frame is the start frame of the the first word and the end frame is the end frame of the last word covered by the concept. On each edge, we have an acoustic score , which is the multiplication of the acoustic scores of the words spanned by the concept, and the phrase probability (determined from the concept's SCFG) . One can compute the concept posterior probabilities using the forward backward algorithm described above. On a concept graph, the emmision prob-

ability is the acoustic probability multiplied by the phrase probability. The transition probabilities are computed using the trigram concept LM. The corresponding forward backward equations are

$$\alpha(E_{c_i,s}^{c_{i+1},e+1}) = p(o_s^e|c_i) \cdot \sum_{c_{i-1},s'} \alpha(E_{c_{i-1},s'}^{c_i,s}) \cdot p_S(c_{i+1}|c_{i-1},c_i) \quad (5)$$

$$\beta(E_{c_i,s}^{c_{i+1},e+1}) = \sum_{c_{i+2},e'+1} \beta(E_{c_{i+1},e+1}^{c_{i+2},e'+1}) \cdot p(o_{e+1}^{e'}|c_{i+1}) \cdot p_S(c_{i+2}/c_i,c_{i+1}) \quad (6)$$

$$p(E_{c_i,s}^{c_{i+1},e+1}) = \frac{\alpha(E_{c_i,s}^{c_{i+1},e+1}) \cdot \beta(E_{c_i,s}^{c_{i+1},e+1})}{p(o_1^T)} \quad (7)$$

where $p_S(.)$ is the probability conditioned on the dialog context, denoted by S,

$$p(o_s^e|c_i) = p(o_s^e|w_{i,1},\cdots w_{i,L_i})p(w_{i,1},\cdots,w_{i,L_i}|c_i) = \prod_{l=1}^{L_i} p(o_{s_l}^{e_l}|w_{i,l})p(r_{i,l}/c_i) \quad (8)$$

, $s_1 = s$, $e_{L_i} = e$, $r_{i,l}$ is the arc of c_i's RTN labeled by the word $w_{i,l}$ and $w_{i,1},\cdots,w_{i,L_i}$ is the L_i-word phrase covered by the concept c_i.

For the sake of simplicity we excluded two things in the derivation of the forward/backward expressions. One is the downscaling of acoustic scores and the other is the interpolation of concept-based SLM with a word/class based SLM. The former is required due to the fact that acoustic scores and LM probabilities have entirely different dynamic ranges. To avoid the domination of summations by a small number of terms, the downscaling of the acoustic scores has been found very useful [7]. Our observations during experiments were on the same line. The interpolation is required to take the advantage of the complementary nature of two different SLMs. It is possible that the interpolation partially recovers the loss from context free assumption. The interpolation is performed at the concept/phrase level. That is, the term

$$p(w_{i,1},\cdots,w_{i,L_i}|c_i) \cdot p_S(c_i|c_{i-2},c_{i-1}) \quad (9)$$

in the expressions above is replaced by

$$(p(w_{i,1},\cdots,w_{i,L_i}|c_i) \cdot p_S(c_i|c_{i-2},c_{i-1}))^{\lambda} \cdot \prod_{l=1}^{L_i} p(w_{i,l}|w_{i,l-2},w_{i,l-1})^{1-\lambda} \quad (10)$$

where $w_{i,0}$ and $w_{i,-1}$ are taken from preceding concepts. That is, $w_{i,0} = w_{i-1,L_{i-1}}$ and $w_{i,-1} = w_{i-1,L_{i-1}-1}$, if $L_{i-1} > 1$. Otherwise, $w_{i,-1} = w_{i-2,L_{i-2}}$. Note that the interpolation is log-linear and λ is the interpolation weight. This interpolation method is selected for two reasons. First, it is easier to implement since the actual implementation deals with the logarithm of probabilities. Second, its performance has been found better than the linear interpolation [12, 13].

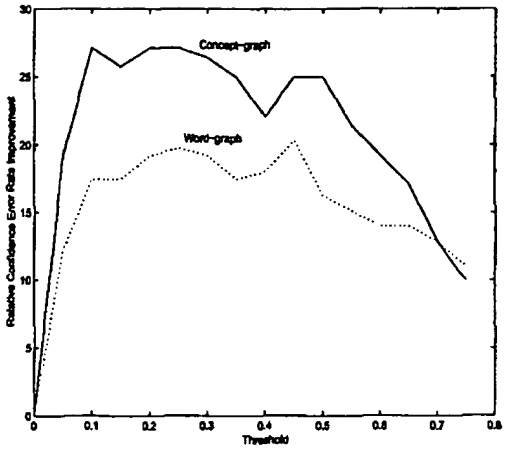


Fig. 2. Relative CER improvement with respect to threshold

Let C_{w_i} be the set of concepts that include the word event $[w_i, s, e]$. Using the concept posteriors we calculate the confidence of w_i as

$$C(w_i) = \sum_{c_i \in C_{w_i}} \sum_{c_{i+1},e+1} p(E_{c_i,s}^{c_{i+1},e+1}) \quad (11)$$

4. EXPERIMENTAL RESULTS

We present experimental results on CU Communicator data. The CU communicator system is a dialog system used for flight, hotel and rental car reservations [14]. The data that we experimented with was collected during National Institute of Standards (NIST) 2000 evaluation. It is from a total of 72 calls. The number of female and male callers are 44 and 28, respectively. A total of 1264 sentences were used in the experiments. Of these, 450 sentences were used to optimize scaling, interpolation factors and tagging thresholds. The final results are on the rest of the data.

We used confidence error rates (CERs) and receiver operating characteristics (ROC) curves to evaluate the confidence measures. The CER is defined as

$$CER = \frac{\text{\# incorrectly tagged words}}{\text{\# recognized words}}$$

The baseline CER is obtained assuming that all recognized words are tagged as correct. This is equivalent to the summation of insertions and substitutions divided by the number of recognized words. The ROC curve is the plot of the correct rejection with respect to false rejection. The correct rejection is tagging the incorrect word as incorrect and the false rejection is tagging the correct word as incorrect.

Figure 2 shows relative CER performance improvements over the baseline CER for both word graph and concept graph based word posteriors. It is plotted with respect to the threshold, as the

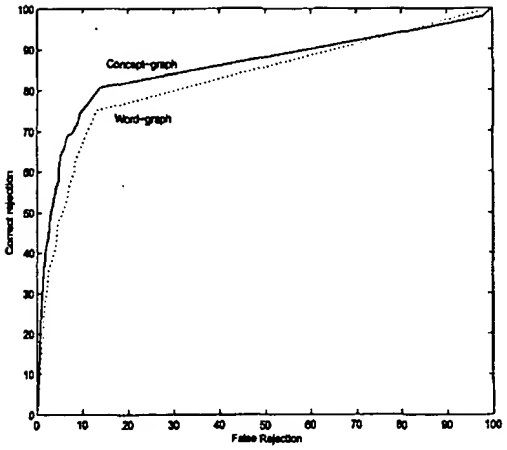


Fig. 3. Receiver operating characteristics (ROC) curve

Table 1. CER and correct rejection (CR) results at 5% false rejection (FR)

Method	Baseline	CER	Reduction	CR
word graph	14.4%	11.8%	18.1%	48.9%
concept graph	11.9%	8.8%	26.1%	63.2%

CER strongly depends on the choice of the tagging threshold. The performances are compared relative to the baseline since two recognizers have different operating points. Figure 3 shows the ROC curve. Both illustrate that the concept graph method performs better.

Table 1, gives baseline CERs, the CERs after confidence annotation, relative reduction in CERs and correct rejection (CR) results at 5% false rejection (FR). All figures clearly show the better performance of the confidence measure computed on concept graphs.

5. CONCLUSIONS

We have presented a confidence measure based on concept/word posterior probabilities computed on concept graphs. We have shown that it outperforms a similar confidence measure based on word graphs. We believe that the improvement is due to the incorporation of higher level knowledge sources (syntactic and semantic constraints) into the computation of posterior probabilities. We plan to use this confidence measure for rescoring the concept graph to improve recognition performance. In addition, the use of this confidence measure in MLLR adaptation is worthy of future research.

6. REFERENCES

[1] S. Cox and R. Rose, "Confidence measures for the switchboard database," in *International Conference of Acoustics, Speech, and Signal Processing*, May 1996, pp. 511–514.

[2] L. Chase, "Word and acoustic confidence annotation for large vocabulary speech recognition," in *Fifth European Conf. on Speech Communication and Technology*, Rhodes, Greece, September 1997, pp. 815–818.

[3] M. Weintraub, F. Beaufays, Z. Rivlin, Y. Konig, and A. Stolcke, "Neural network based measures of confidence for word recognition," in *International Conference of Acoustics, Speech, and Signal Processing*, Munich, Germany, April 1997, pp. 887–990.

[4] T. Kemp and T. Schaaf, "Estimating confidence using word lattices," in *Fifth European Conf. on Speech Communication and Technology*, Rhodes, Greece, September 1997, pp. 827–830.

[5] F. Wessel, K. Macherey, and R.Schluter, "Using word probabilities as confidence measures," in *International Conference of Acoustics, Speech, and Signal Processing*, Seattle, WA, May 1998, pp. 225–228.

[6] F. Wessel, K. Macherey, and H. Ney, "A comparison of word graph and n-best list based confidence measures," in *International Conference of Acoustics, Speech, and Signal Processing*, Seattle, WA, May 1998, pp. 225–228.

[7] F. Wesse, R. Schluter, K. Macherey, and H. Ney, "Confidence measures for large vocabulary continuous speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 3, pp. 288–298, March 2001.

[8] R. San-Segundo, B. Pellom, K. Hacioglu, W. Ward, and J.M. Pardo, "Confidence measures for dialogue systems," in *International Conference on Acoustics, Speech, and Signal Processing*, Salt Lake City, Utah, May 2001.

[9] R. Zhang and A. Rudnicky, "Word level confidence annotation using combination of features," in *Seventh European Conf. on Speech Communication and Technology*, Aalborg, Denmark, September 2001, pp. 2105–2108.

[10] K. Hacioglu and W. Ward, "A word graph interface for a flexible concept based speech understanding framework," in *Seventh European Conf. on Speech Communication and Technology*, Aalborg, Denmark, September 2001, pp. 1775–1778.

[11] M. K. Ravishankar, *Efficient Algorithms for Speech Recognition*, Ph.D. thesis, Carnegie Mellon University, 1996.

[12] D. Klakow, "Log-linear interpolation of language models," in *5-th International Conference on Spoken Language Processing*, Sydney, Australia, 1998, pp. 1695–1699.

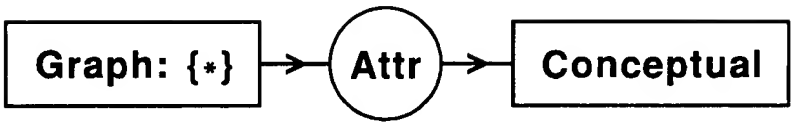
[13] K. Hacioglu and W. Ward, "On combining language models: Oracle approach," in *First International Conference on Human Language Technology Research*, San Diego, California,, March 18-21 2001.

[14] W. Ward and B. Pellom, "The CU communicator system," in *IEEE Workshop on Automatic Speech Recognition and Understanding*, Keystone, Colorado, 1999.

Conceptual Graphs

Documentation
CG Standard
CG Examples
Peirce's EGs
Bibliography

CG Tools
CGWorld
CharGer
Cogito
Notio
Prolog+CG
WebKB



Conceptual graphs (CGs) are a system of logic based on the existential graphs of Charles Sanders Peirce and the semantic networks of artificial intelligence. They express meaning in a form that is logically precise, humanly readable, and computationally tractable. With a direct mapping to language, conceptual graphs serve as an intermediate language for translating computer-oriented formalisms to and from natural languages. With their graphic representation, they serve as a readable, but formal design and specification language. CGs have been implemented in a variety of projects for information retrieval, database design, expert systems, and natural language processing.

For examples of conceptual graphs and their translations to predicate calculus and the Knowledge Interchange Format (KIF):

- **Examples using the standard HTML 4.0 symbol definitions**

- **Examples using the symbols font in MS Windows** (suitable for older versions of Navigator or Internet Explorer).

Other Links
ICCS'2001
KR Book
Ontology
Logic

For the conceptual graph standard:

NEW **Working draft of the proposed ISO standard**

For a bibliography of books and conference proceedings about conceptual graphs:

- **CG bibliography**

For links to other information about conceptual graphs, including conferences and tools:

- **CG home page at the University of Alabama at Hunstville**

Send e-mail
CG List
John F. Sowa

To subscribe to the conceptual graph mailing list, send the following message to majordomo@cs.uah.edu.

```
To: majordomo@cs.uah.edu
```

```
Subject: subscribe cg
subscribe cg
end
```

Last Modified: *07/03/2001 23:16:13*

Conceptual Graph Examples

Conceptual graphs are formally defined in an abstract syntax that is independent of any notation, but the formalism can be represented in several different concrete notations. This document illustrates CGs by means of examples represented in the graphical *display form* (DF), the formally defined *conceptual graph interchange form* (CGIF), and the compact, but readable *linear form* (LF). Every CG is represented in each of these three forms and is translated to a logically equivalent representation in predicate calculus and in the Knowledge Interchange Format (KIF). For the formal definition of conceptual graphs and the various notations for representing them, see the draft proposed American National Standard. For examples of an English-like notation for representing logic, see the web page on controlled English.

List of Examples

Following are some sample sentences that are represented in each of the notations: CGs, KIF, and predicate calculus. Click on the sentence to go directly to its representation.

---

1. *A cat is on a mat.*

In the display form (DF), concepts are represented by rectangles: the concept [Cat] represents a instance of a cat, and [Mat] represents an instance of a mat. Conceptual relations are represented by circles or ovals: the conceptual relation (On) relates a cat to a mat. The arcs that link the relations to the concepts are represented by arrows: the first arc has an arrow pointing toward the relation, and the second arc has an arrow pointing away from the relation. If a relation has more than two arcs, the arcs are numbered.

In the linear form (LF), concepts are represented by square brackets instead of boxes, and the conceptual relations are represented by parentheses instead of circles:

```
[Cat]->(On)->[Mat].
```

Both DF and LF are designed for communication with humans or between humans and machines. For communication between machines, the conceptual graph interchange form (CGIF) has a syntax that uses *coreference labels* to represent the arcs:

```
[Cat: *x] [Mat: *y] (On ?x ?y)
```

The symbols *x and *y are called *defining labels*. The matching symbols ?x and ?y are the *bound labels* that indicate references to the same instance of a cat *x* or a mat *y*. To reduce the number of coreference labels, CGIF also permits concepts to be nested inside the relation nodes:

```
(On [Cat] [Mat])
```

The display form in Figure 1 represents the abstract CG most directly. All the variations of LF and CGIF represent stylistically different, but logically equivalent ways of linearizing the same abstract graph. All these variations are accommodated by the LF grammar and the CGIF grammar, which are defined in the CG standard.

CGIF is intended for transfer between computer systems that use CGs as their internal representation. For communication with systems that use other internal representations, CGIF can be translated to another logic-based formalism called the Knowledge Interchange Format (KIF):

```
(exists ((?x Cat) (?y Mat)) (On ?x ?y))
```

Although DF, LF, CGIF, and KIF look very different, their semantics is defined by the same logical foundations. They can all be translated to a statement of the following form in typed predicate calculus:

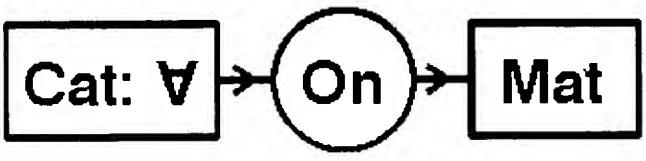
```
(∃x:Cat)(∃y:Mat)on(x,y).
```

Any statement expressed in any one of these notations can be automatically translated to a logically equivalent statement in any of the others. Formatting and stylistic information, however, may be lost in translations between DF, LF, CGIF, KIF, and predicate calculus.

---

2. *Every cat is on a mat.*

The default quantifier in a concept is the existential ∃, which is normally represented by a blank. The concept [Cat] without anything in the referent field is logically equivalent to the concept [Cat: ∃], which asserts the proposition that there exists a cat. Other quantifiers, such as the universal ∀, are called *defined quantifiers* because they can be defined in terms of conceptual graphs containing only the default existential. In Figure 2, the concept [Cat: ∀] represents the phrase *every cat*, and the complete CG represents the sentence *Every cat is on a mat.*



In the linear form (LF), the universal quantifier may be represented by the symbol ∀ if it is available. Otherwise, it is represented by the symbol @every. Both of the following CGs are semantically identical:

```
[Cat: ∀]->(On)->[Mat].

[Cat: @every]->(On)->[Mat].
```

Since CGIF is expressible in the 7-bit subset of ASCII or Unicode, the character ∀ must be represented by @every in CGIF:

```
[Cat: @every*x] [Mat: *y] (On ?x ?y)
```

As in Figure 1, CGIF permits concepts to be nested inside the relation nodes:

```
(On [Cat: @every] [Mat])
```

In all these examples, the universal quantifier @every or ∀ includes the default existential quantifiers in the same context within its scope. The scope is enforced by the definition of the quantifier @every in terms of the existential. When the definition is expanded, the CG in Figure 2 is expanded to a CG that can be represented by the following LF graph:

```
~[ [Cat: *x]
   ~[ [?x]->(On)->[Mat] ]].
```

Literally, this CG may be read *It is false that there exists a cat x that is not on a mat.* An alternate reading treats the two nested negations as an implication: *If there exists a cat x, then x is on a mat.* Following is a CGIF representation of the expanded CG:

```
~[ [Cat: *x]  ~[ (On ?x [Mat]) ]].
```

All of these forms are logically equivalent to the original CG in Figure 2.

Since KIF has a universal quantifier, it is not necessary to expand the defined quantifier @every before translating a CG to KIF. Following is the KIF translation of Figure 2:

```
(forall ((?x Cat)) (exists ((?y Mat)) (On ?x ?y)))
```

This KIF statement is logically equivalent to the KIF statement that results from translating the expanded CG:

```
(not (exists ((?x Cat)) (not (exists ((?y Mat)) (On ?x ?y)))))
```

The original CG in Figure 2 may be represented by the following formula in typed predicate calculus:

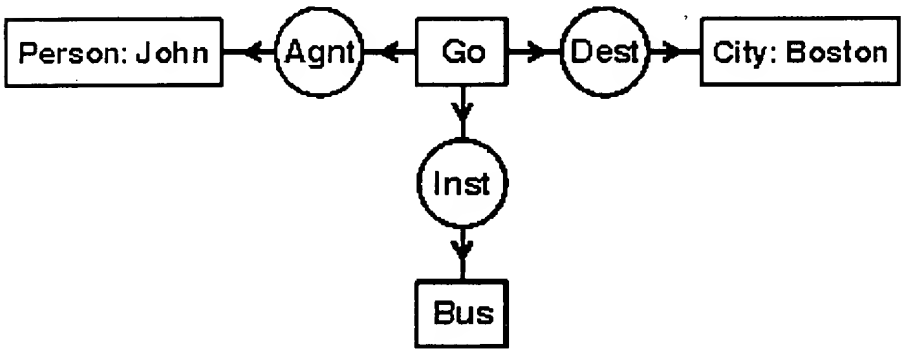
```
(∀x:Cat)(∃y:Mat)on(x,y).
```

This formula is logically equivalent to the formula that represents the expanded CG:

```
~(∃x:Cat)~(∃y:Mat)on(x,y).
```

3. *John is going to Boston by bus.*

Figure 3 shows a conceptual graph with four concepts: [Go], [Person: John], [City: Boston], and [Bus]. It has three concdeptual relations: (Agnt) relates [Go] to the agent John, (Dest) relates [Go] to the destination Boston, and (Inst) relates [Go] to the instrument bus.



Since the concept [Go] is attached to three conceptual relations, the linear form cannot be drawn in a straight line, as in Figure 1. Instead, a hyphen at the end of the first line indicates that the relations attached to [Go] are continued on subsequent lines.

```
[Go]-
   (Agnt)->[Person: John]
   (Dest)->[City: Boston]
   (Inst)->[Bus].
```

This example resembles frame notation, but LF also permits coreference labels to represent the cross references needed to represent arbitrary graphs.

In the following CGIF representation for Figure 3, each concept has its own defining label:

```
[Go: *x] [Person: John *y] [City: Boston *z] [Bus: *w]
   (Agnt ?x ?y) (Dest ?x ?z) (Inst ?x ?z)
```

By nesting some of the concepts inside the relations, the CGIF form can be limited to just a single defining label *x and a bound label ?x inside each relation node:

```
[Go *x] (Agnt ?x [Person: John]) (Dest ?x [City: Boston]) (Inst ?x [Bus])
```

The display form in Figure 3 represents the abstract CG most directly. All the variations of LF and CGIF represent different, but logically equivalent ways of linearizing the same abstract graph.

The version of CGIF that assigns a separate defining label to each concept usually has the most direct mapping to KIF:

```
(exists ((?x Go) (?y Person) (?z City) (?w Bus))
        (and (Name ?y John) (Name ?z Boston)
             (Agnt ?x ?y) (Dest ?x ?z) (Inst ?x ?w)))
```

Following is the corresponding formula in typed predicate calculus:

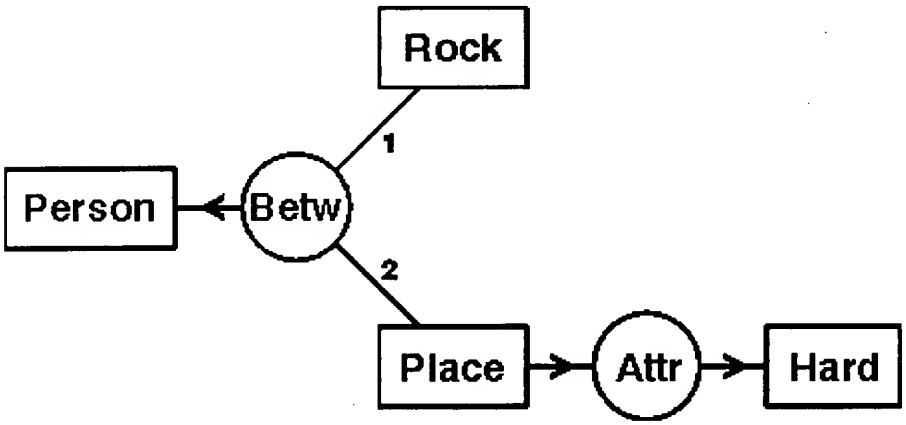
```
(∃x:Go)(∃y:Person)(∃z:City)(∃w:Bus)
   (name(y,'John') ∧ name(z,'Boston') ∧
      agnt(x,y) ∧ dest(x,z) ∧ inst(x,w))
```

For a list of the relations that connect the concepts corresponding to verbs to the concepts of their participants, see the web page on thematic roles.

---

4. *A person is between a rock and a hard place.*

The between relation (Betw) is a triadic relation, whose first two arcs are linked to concepts of entities that occur on either side of the entity represented by the concept linked to the third arc. For a conceptual relation with n arcs, the first n-1 arcs have arrows that point toward the circle, and the n-th or last arc points away.



In LF, Figure 4 may be represented in the following form:

```
[Person]<-(Betw)-
           <-1-[Rock]
           <-2-[Place]->(Attr)->[Hard].
```

The hyphen after the relation indicates that its other arcs are continued on subsequent lines. The two arcs that point towards the relation are numbered 1 and 2. The arc that points away is the last or third arc; the number 3 may be omitted, since it is implied by the outward pointing arrow. For monadic relations, both the number 1 and the arrow pointing towards the circle are optional. For dyadic relations, the arcs are either numbered 1 and 2, or the first arc points towards the circle and the second arc points away.

CGIF allows any number of concepts to be nested inside the relations:

```
(Betw [Rock] [Place *x] [Person])  (Attr ?x [Hard])
```

For relations with more than 2 arcs, CGIF notation is more compact than the multiline LF notation. Therefore, most LF implementations allow CGIF notation to be mixed with the arrow notation:

```
[Place: *x]->(Attr)->[Hard]  (Betw [Rock] ?x [Person]).
```

In the CG standard, the only notation that must be standardized is CGIF, since every implementation must recognize exactly the same forms. The LF and DF notations are specified only in an informative annex to the CG standard. Therefore, implementers may experiment with different variations in an attempt to improve readability or convenience. Nevertheless, too much variation may make it harder for human readers to switch from one version to another.

Following is the KIF representation:

```
(exists ((?x person) (?y rock) (?z place) (?w hard))
        (and (betw ?y ?z ?x) (attr ?z ?w)))
```

And following is the corresponding formula in predicate calculus:

```
(∃x:Person)(∃y:Rock)(∃z:Place)(∃w:Hard)
     (betw(y,z,x) ∧ attr(z,w))
```

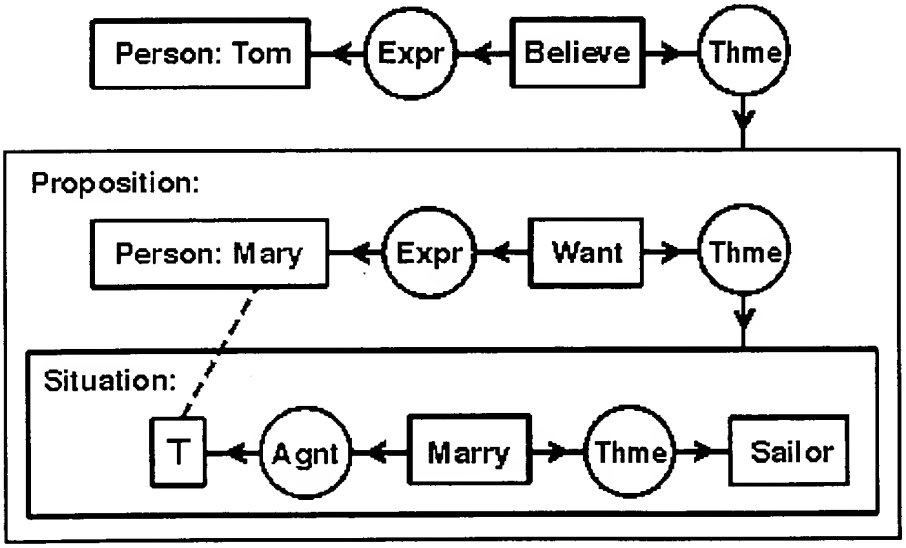
To emphasize the correspondence between CGs and KIF, it is possible to write CGIF in a form in which the concept nodes with their quantifiers and coreference labels precede the relation nodes:

```
[Person *x] [Rock *y] [Place *z] [Hard *w]
     (betw ?y ?z ?x) (attr ?z ?w)
```

---

5. *Tom believes that Mary wants to marry a sailor.*

A context is a concept with a nested conceptual graph that describes the referent. In Figure 4, the concept of type Proposition is a context that describes a proposition that Tom believes. Inside that context is another context of type Situation, which describes a situation that Tom believes Mary wants. The resulting CG represents the sentence *Tom believes that Mary wants to marry a sailor.*



In Figure 5, Tom is the experiencer (Expr) of the concept [Believe], which is linked by the theme relation (Thme) to a proposition that Tom believes. The proposition box contains another conceptual graph, which says that Mary is the experiencer of [Want], which has as theme a situation that Mary

hopes will come to pass. That situation is described by another nested graph, which says that Mary (represented by the concept [⊤]) marries a sailor. The dotted line, called a *coreference link*, shows that the concept [⊤] in the situation box refers to the same individual as the concept [Person: Mary] in the proposition box. Following is the corresponding linear form:

```
[Person: Tom]<-(Expr)<-[Believe]->(Thme)-
   [Proposition:  [Person: Mary *x]<-(Expr)<-[Want]->(Thme)-
      [Situation:  [?x]<-(Agnt)<-[Marry]->(Thme)->[Sailor] ]].
```

Both the display form and the linear form follow the same rules for the scope of quantifiers. The part of the graph outside the nested contexts contains three concepts: [Person: Tom], [Believe], and the proposition that Tom believes. That part of the graph asserts information that is assumed to be true of the real world.

Inside the proposition box are three more concepts: [Person: Mary], [Want], and the situation that Mary wants. Since those three are only asserted within the context of Tom's belief, the graph does not imply that they must exist in the real world. Since Mary is a named individual, one might give her the benefit of the doubt and assume that she also exists; but her desire and the situation she supposedly desires exist in the context of Tom's belief. If his belief is false, the referents of those concepts might not exist in the real world. Inside the context of the desired situation are the concepts [Marry] and [Sailor], whose referents exist within the scope of Mary's desire, which itself exists only within the scope of Tom's belief.

Following is the CGIF representation for Figure 5:

```
[Person: *x1 'Tom'] [Believe *x2] (Expr ?x2 ?x1)
   (Thme ?x2 [Proposition:
      [Person: *x3 'Mary'] [Want *x4] (Expr ?x4 ?x3)
         (Thme ?x4 [Situation:
            [Marry *x5] (Agnt ?x5 ?x3) (Thme ?x5 [Sailor]) ]) ])
```

Following is the KIF statement:

```
(exists ((?x1 person) (?x2 believe))
   (and (expr ?x2 ?x1)
        (thme ?x2
           (exists ((?x3 person) (?x4 want) (?x8 situation))
              (and (name ?x3 'Mary) (expr ?x4 ?x3) (thme ?x4 ?x8)
                   (dscr ?x8 (exists ((?x5 marry) (?x6 sailor))
                                (and (Agnt ?x5 ?x3) (Thme ?x5 ?x6)))))))))
```

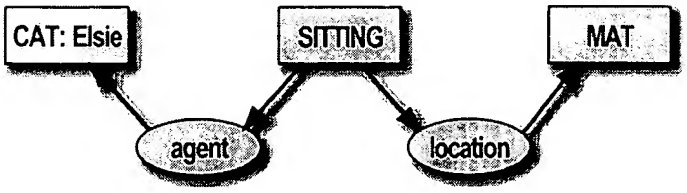
Following is the predicate calculus formula:

```
(∃x1:Person)(∃x2:Believe)(expr(x1,x2) ∧
   thme(x2, (∃x3:Person)(∃x4:Want)(∃x8:Situation)
              (name(x3,'Mary') ∧ expr(x4,x3) ∧ thme(x4,x8) ∧
               dscr(x8, (∃x5:Marry)(∃x6:Sailor)
                          (agnt(x5,x3) ∧ thme(x5,x6)))))))
```

For further discussion of contexts and their representation in conceptual graphs and predicate calculus, see Chapter 5 of the book *Knowledge Representation*.

Last Modified: *07/03/2001 23:13:33*

A World of Conceptual Graphs



Conceptual graphs (CGs) are a system of logic based on the existential graphs of Charles Sanders Peirce and the semantic networks of artificial intelligence. They express meaning in a form that is logically precise, humanly readable, and computationally tractable. With their direct mapping to language, conceptual graphs serve as an intermediate language for translating computer-oriented formalisms to and from natural languages. With their graphic representation, they serve as a readable, but formal design and specification language. CGs have been implemented in a variety of projects for information retrieval, database design, expert systems, and natural language processing.

Introductions to Conceptual Graphs

- Simon Polovina and John Heaton, "An Introduction to Conceptual Graphs," AI Expert, pp. 36-43, 1992.
- John F. Sowa, Information Processing in Mind and Machine. Reading, MA: Addison-Wesley Publ., 1984.
- John F. Sowa, "Conceptual Graphs Summary," in Conceptual Structures: Current Research and Practice, P. Eklund, T. Nagle, J. Nagle, and L. Gerholz, eds., Ellis Horwood, 1992, pp. 3-52.
- John Sowa's brief summary.
- John Sowa, Knowledge Representation : Logical, Philosophical, and Computational Foundations, is not limited to conceptual graphs, but provides broad coverage of the entire field. It is available from Barnes and Noble books, Amazon.com etc.
- Aalborg University's Department of Communication has developed an excellent Online Conceptual Graphs course

Conceptual Graphs Standard Notation

At the moment, the standard is in draft form, but we expect the process to be completed in the next few months. In the meantime, a copy of the (nearly finalized) standard can be found below:

For examples of conceptual graphs and their translations to predicate calculus and the Knowledge Interchange Format (KIF):

- Examples using the standard HTML 4.0 symbol definitions
- Examples using the symbols font in MS Windows (suitable for older versions of Navigator or Internet Explorer).

For the draft proposed American National Standard for conceptual graphs:

- CG dpANS using the standard HTML 4.0 symbol definitions

- CG dpANS using the symbols font in MS Windows (suitable for older versions of Navigator or Internet Explorer)

Conceptual Graph Bibliographies

Bibliography from the CG standard

John Sowa's bibliography page

Conceptual Graphs Conferences

International Conference on Conceptual Structures

13th International Conference on Conceptual Structures (ICCS 2005)
July 18-22, 2005, Kassel. Germany. For information write to: stumme@cs.uni-kassel.de

Previous conferences:

12th International Conference on Conceptual Structures (ICCS 2004)
July 19-23, 2004, Huntsville, Alabama, U.S.A.

11th International Conference on Conceptual Structures (ICCS 2003)
July 21-25, 2003, Dresden, Germany

Tenth International Conference on Conceptual Structures (ICCS 2002)
July 15 - 19, 2002, Borovets, Bulgaria.

Ninth International Conference on Conceptual Structures (ICCS 2001)
July 30 - Aug. 3, 2001, Palo Alto, California, U.S.A.

Eighth International Conference on Conceptual Structures (ICCS 2000)
August 14-18, 2000, Darmstadt, Germany

Conceptual Graphs Email List

An archive of the list messages (since September 2000) can be found at http://mars.virtual-earth.de/pipermail/cg/ (thanks to Mathias Picker)

Harry Delugach, Conceptual Graphs Mailing List Administrator, delugach@cs.uah.edu .

To POST to the list, send mail to cg@cs.uah.edu (you must first join the list)

To JOIN (receive the postings), send mail as follows or click here. Be sure the "From" email address is the one you want!

```
To: majordomo@cs.uah.edu
   Subject: <ignored>
   subscribe cg
   end
```

To UNSUBSCRIBE (get off the list), send mail as follows or click here. Be sure to include the exact email address which is currently subscribed.

```
To: majordomo@cs.uah.edu
   Subject: <ignored>
   unsubscribe cg <your subscribed email address>
```

end

Research Groups and Projects

Universite Laval, Quebec City, Canada
Cognitive Informatics Laboratory

Virginia Polytechnic and State University (Virginia Tech)
Automatic Design Research Group

Washington State University
Conceptual Knowledge Markup Language Project
Ontology Links

University of California, Santa Cruz
Fast Conceptual Graph Retrieval and Question Answering

SIAD
Development and use of CG tools
in knowledge base and expert systems, natural language processing, Case Based Reasoning and learning, Intelligent Tutoring Systems and Multi-Agent Systems.

Aalborg University - Department of Communication
Online Conceptual Graphs course

Tools

CharGer, a prototype conceptual graph editor developed the University of Alabama in Huntsville, free to noncommercial use, runs under Java

Notio - an API specification for a set of Java classes designed to provide an implementation-independent interface for manipulating Conceptual Graphs, GNU license

WebKB - tools for information retrieval and knowledge representation, availability??

CG Mars Lander - fast conceptual graph retrieval and question answering tool, available for joint development and industrial funding.

Prolog+CG - an object-oriented extension of PROLOG, based on CG. CG (both simple and compound) is a basic data structure, like term. PROLOG+CG is implemented with Java 2.

CoGITaNT - several useful utilities: a set of library routines in C++ for conceptual modeling, some knowledge bases in conceptual graphs, and an XML specification for CGXML. en francais.

There's also an older list of Conceptual Graph Tools

Individual Researchers

Kalina Bontcheva	University of Sheffield, UK	Use of conceptual graphs for natural language generation	kalina@dcs.shef.ac.uk http://www.dcs.shef.ac.uk/~kalina
William C. Burkett		Data Model Design and Quality, Database schema design	wcb@usc.edu, wburkett@pdit.com
Tru Hoang Cao	Ho Chi Minh Univ. Technology, Vietnam	Uncertain and imprecise knowledge representation and reasoning	tru@dit.hcmut.edu.vn http://www.dit.hcmut.edu.vn/~tru

Michel Chein	LIRMM, Montpellier	CORALI (COnceptual gRAph at LIrmm)	chein@lirmm.fr http://www.lirmm.fr/~cogito
Aldo de Moor	Infolab, Tilburg Univ., The Netherlands	Application of conceptual graphs to information system specification	http://infolab.kub.nl/people/ademoor
delugach@cs.uah.edu Harry Delugach	Univ. Alabama in Huntsville, U.S.A.	CGs in software requirements CGs in knowledge acquisition	http://www.cs.uah.edu/~delugach
Bob Levinson Gil Fuchs	Univ. California, Santa Cruz, U.S.A.	Fast conceptual graph retrieval and question answering	
Vladimir N. Polyakov		Natural language processing	http://www.geocities.com/SiliconValley/Campus/7926/Polyakov/Polyakov.htm
John F. Sowa	U.S.A.	Knowledge representation and natural language semantics	http://www.bestweb.net/~sowa/direct
David Cox	U.S.A.	Explanation by pattern: FLIPP	http://www.flipp-explainers.org

(if you'd like to be listed here, please notify the maintainer delugach@cs.uah.edu)

This page maintained by delugach@cs.uah.edu and was last changed on November 4, 2004 .